**Multi Agent RL**

Algorithms where multiple small tasks are accomplished by agents in a shared environment to complete one bigger task. Each agent learns to make decisions based on its observation rewards and behavior of other agents in some cases.

There are two or more agents involved and the interaction between them is cooperative, competitional or mixed.

Type of Interactions

1. Cooperative

Agents work together to accomplish common goals

1. Competitional

Agents have opposing goals

1. Mixed

Mixed between cooperation and competition

Key Challenges

1. Non-Stationarity

With changes in other agent’s behavior, the shared environment becomes more unpredictable

1. Credit Assignment

It’s hard to determine which agent had higher contribution in group reward setting while contributing to single goal

1. Scalability

As the agents grow joint action space grows exponentially

1. Partial Observability

Agent may lack full visibility of the environment or each other

**Meta-RL**

Meta RL is a field where agents learn how to quickly learn new tasks. Instead of taring for a single task, Meta-RL trains the agent to adapt quickly to new tasks using a small amount of data or experience.

Difference between traditional RL and Meta-RL

Traditional RL

* An agent trains for millions of steps on one task
* If the task changes (even slightly), training must start from scratch

Meta-RL

* The agent it trained on distribution of tasks
* At test time, it can quickly adapt to new but related tasks

How it works

1. Gradient-Based Meta-RL (Model-Agnostic Meta-Learning)
2. Recurrent Meta-RL (RNN-based)

Meta-RL Structure

* Meta Training Phase

Training many tasks from task distribution

Learn a policy or algorithm that can adapt

* Meta-Testing Phase

Give new task, it adapts quickly using only a few episodes

Benefits

1. Sample efficiency
2. Generalization
3. Task adaptation

Challenges

1. Require many diverse training tasks
2. Computationally intensive
3. Defining task distribution can be tricky

**Contextual Bandits**

Agents learn to choose the best action for a given context to maximize the rewards over time.

Its balance between Exploration and Exploitation

* Exploration: Try new actions to learn
* Exploitation: Use bets known action for a context

How it works

Context 🡪 Action 🡪 Reward

1. Agents see the context
2. Chooses an action
3. Gets reward
4. Updates the strategy for future similar contexts

Common Algorithms

1. Epsilon-Greedy
2. LinUCB
3. Thompson Sampling
4. Neural Bandits

Benefits

* Fast to train, easy to update
* Doesn’t need full environment modeling like RL
* Strong for personalized experiences

Challenges

* No long-term planning
* Assuming action doesn’t affect future context
* May struggle with high dimensional contexts

**Why Contextual Bandit > Meta-RL in our case?**

1. Sample and effective for Policy Selection
2. Fast Decision making
3. Easier to Train and Debug
4. Integrates well with Sentiment + Macro Context